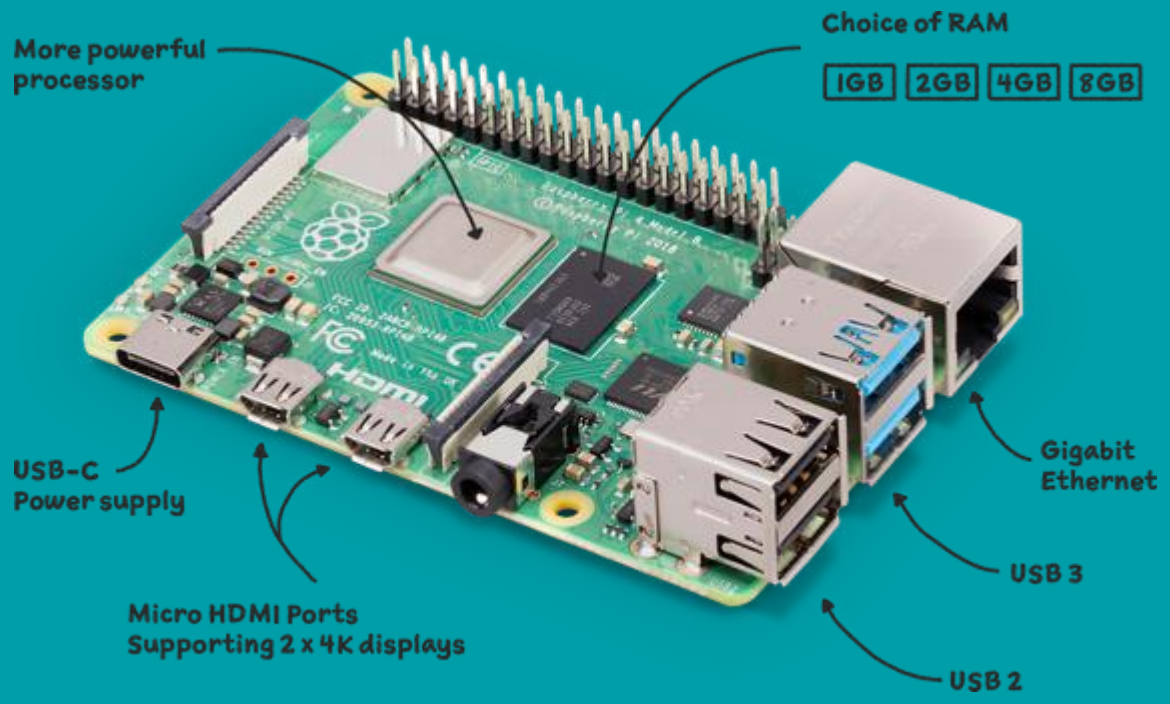


2. GPIO 활용

이영주
young.kopo@gmail.com

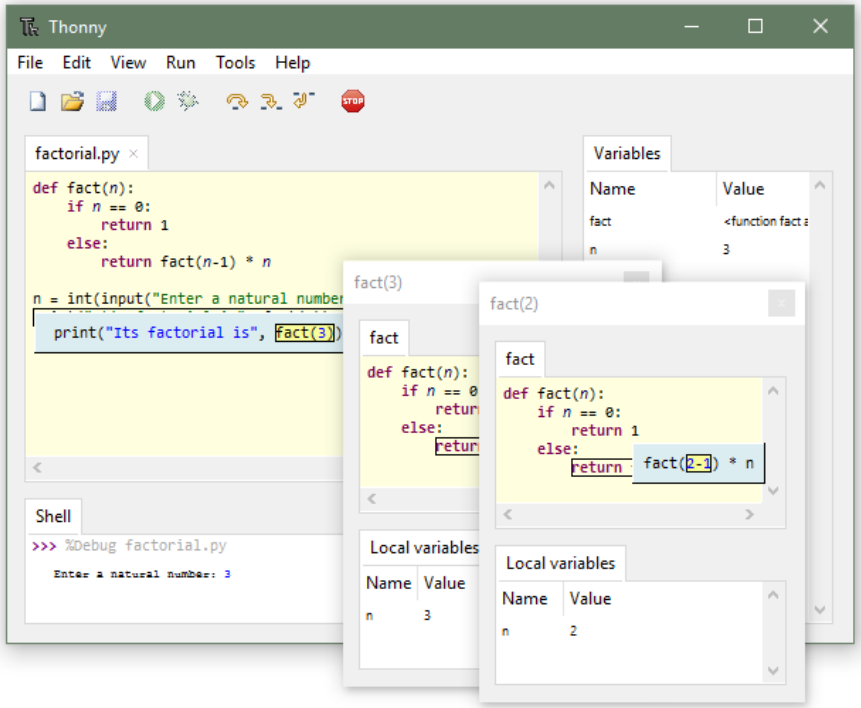
한국폴리텍대학

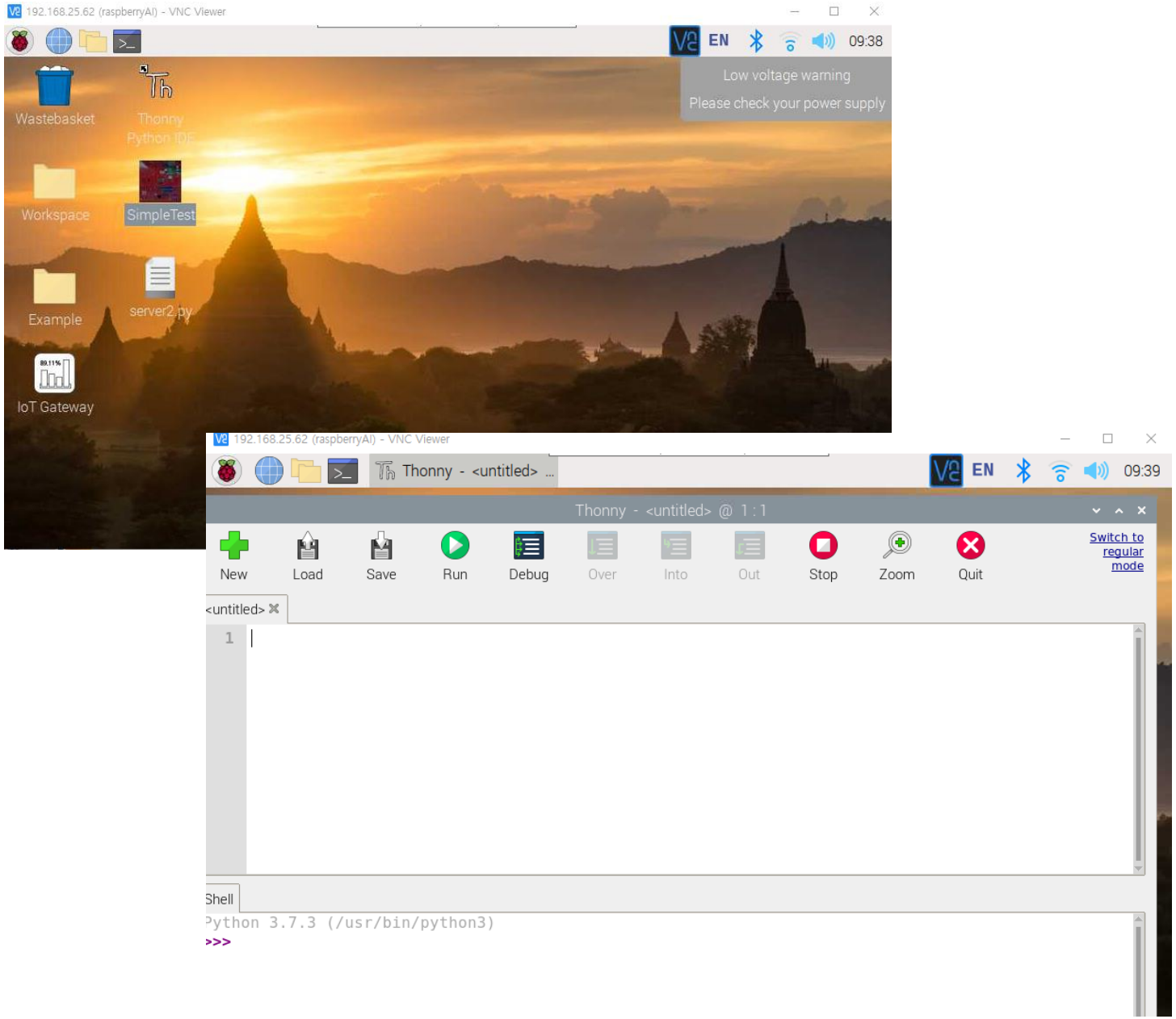
개발도구(Thonny Python IDE)

Thonny 4 is dedicated to Ukraine fighting the Russian invasion.
UA Please support Ukraine! UA

Thonny
Python IDE for beginners

Download version 4.0.2 for
Windows • Mac • Linux





Rpi.GPIO 사용

라즈베리파이 GPIO 제어용 파이썬 모듈 (기본 내장)

Search projects

Help Sponsors Log in Register

RPi.GPIO 0.7.1

```
pip install RPi.GPIO
```

Latest version

Released: Feb 7, 2022

A module to control Raspberry Pi GPIO channels

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Project description

This package provides a Python module to control the GPIO on a Raspberry Pi.

Note that this module is unsuitable for real-time or timing critical applications. This is because you can not predict when Python will be busy garbage collecting. It also runs under the Linux kernel which is not suitable for real time applications - it is multitasking O/S and another process may be given priority over the CPU, causing jitter in your program. If you are after true real-time performance and predictability, buy yourself an Arduino http://www.arduino.cc !

Note that the current release does not support SPI, I2C, hardware PWM or serial functionality on the RPi yet. This is planned for the near future - watch this space! One-wire functionality is also planned.

Although hardware PWM is not available yet, software PWM is available to use on all channels.

For examples and documentation, visit http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

Rpi.GPIO 사용

import RPi.GPIO as GPIO

SOURCEFORGE

Help | Create | Join | Login

Open Source Software | Business Software | Resources

Sync your GitHub Project to SourceForge

Search for software or solutions

Home / Browse / raspberry-gpio-python / Wiki

raspberry-gpio-python Wiki

A Python module to control the GPIO on a Raspberry Pi

Brought to you by: croston

Summary | Files | Reviews | Support | Tickets | Wiki | Code

Search Wiki

Wiki Home

Browse Pages

Browse Labels

Formatting Help

BasicUsage

Authors: Anonymous

RPi.GPIO module basics

Importing the module

To import the RPi.GPIO module:

```
import RPi.GPIO as GPIO
```

By doing it this way, you can refer to it as just GPIO through the rest of your script.

To import the module and check to see if it is successful:

```
try:
    import RPi.GPIO as GPIO
except RuntimeError:
    print("Error importing RPi.GPIO!  This is probably because you need superuser privileges.  You can achieve this by using 'sudo' to run your script")
```

Rpi.GPIO 사용

- 제어핀 모드 설정

```
GPIO.setmode(GPIO.BOARD)
  # or
GPIO.setmode(GPIO.BCM)
```

	Physical Pins				
Function	BCM	pin#	pin#	BCM	Function
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

Rpi.GPIO 사용

- 채널설정(입/출력 설정)

✓ GPIO.setup(channel, GPIO.IN)
✓ GPIO.setup(channel, GPIO.OUT)
✓ GPIO.setup(channel, GPIO.OUT, initial = GPIO.HIGH)

- 입출력 제어

✓ GPIO.input(channel)
✓ GPIO.ouput(channel, state)
➢ state : GPIO.HIGH/1/True, GPIO.LOW/0/False

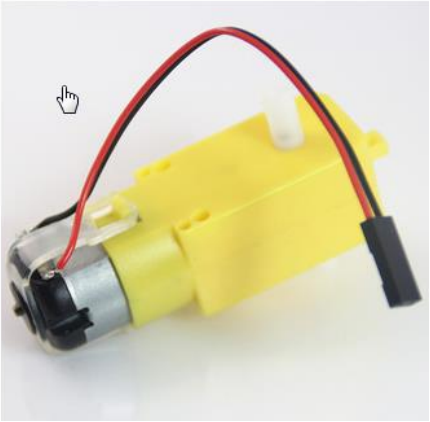
- 종료전 리소스 반납(필수)

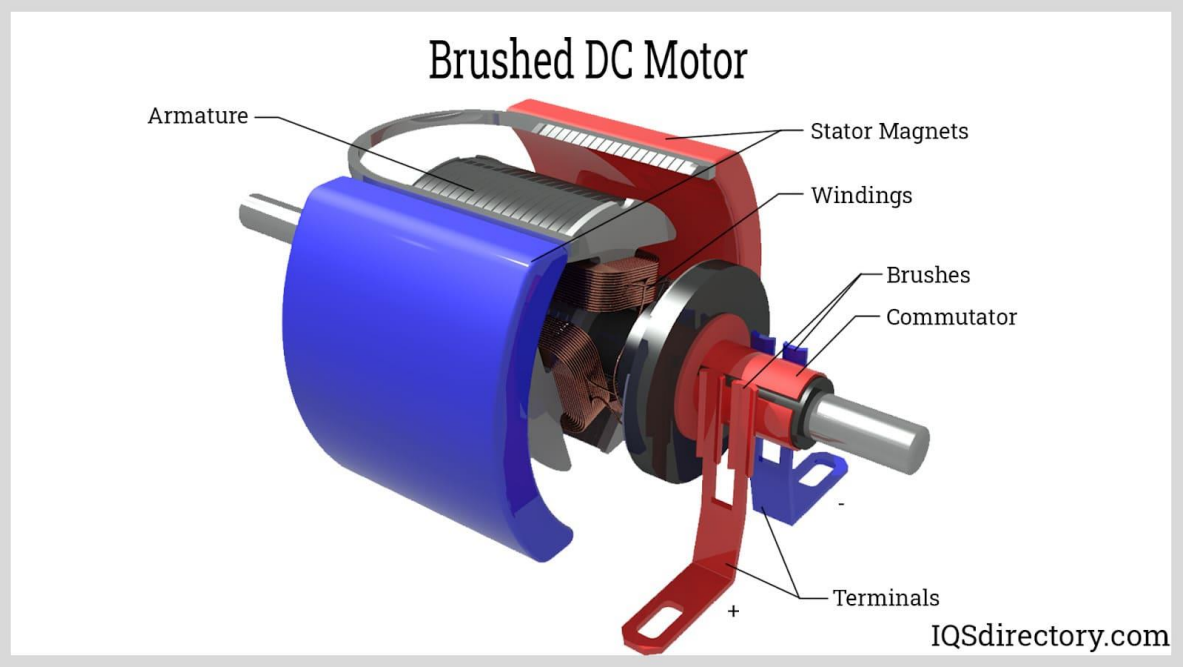
✓ GPIO.cleanup()

Rpi.GPIO 사용 예제

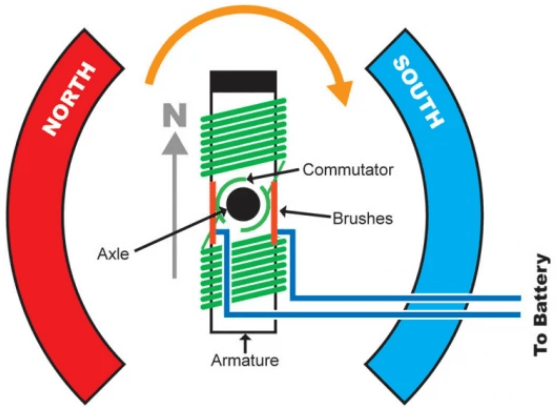
```
# 라이브러리 임포트
import RPi.GPIO as GPIO
 ...
# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.IN)
GPIO.setup(18, GPIO.OUT)
...
# 메인 쓰레드
try:
        while 1:
                button = GPIO.input(12)
                ...
                GPIO.output(18, GPIO.HIGH)
                ...
# 반드시 클린업
finally: GPIO.cleanup()
```

DC모터제어

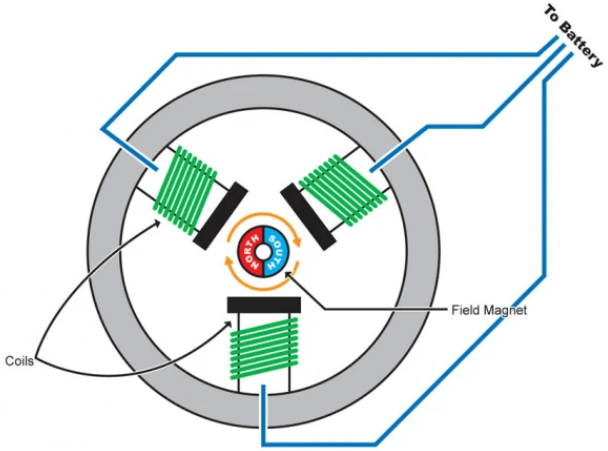




BRUSHED MOTOR



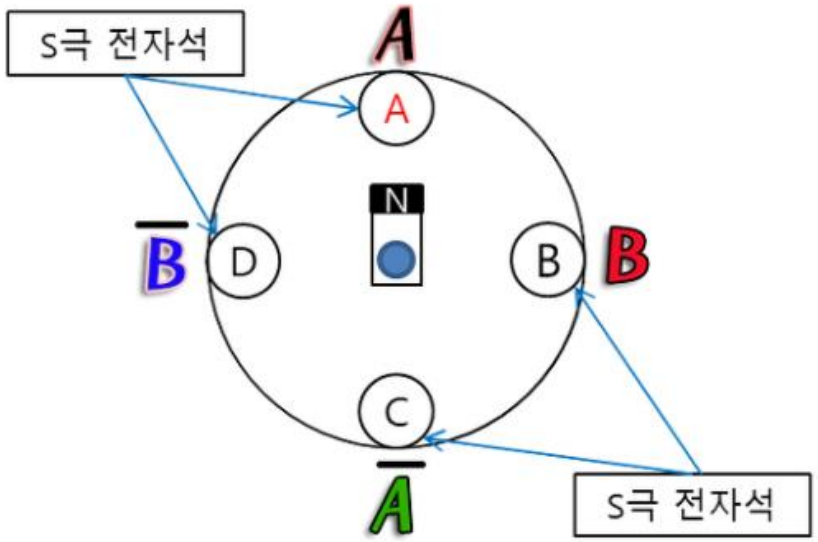
BRUSHLESS MOTOR

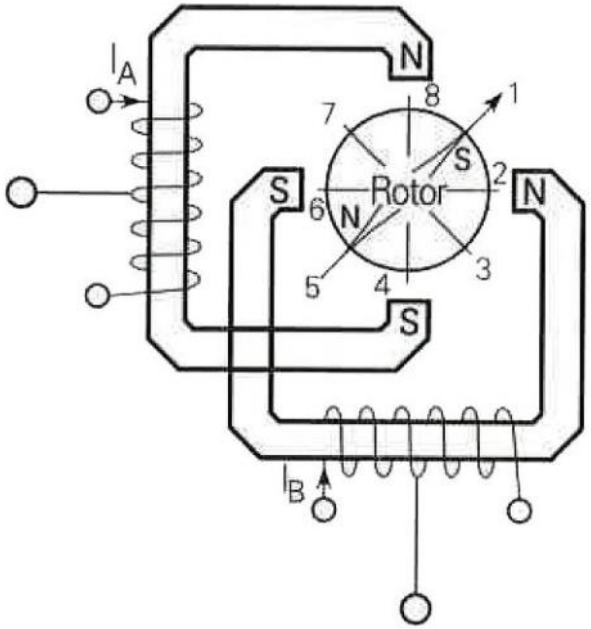


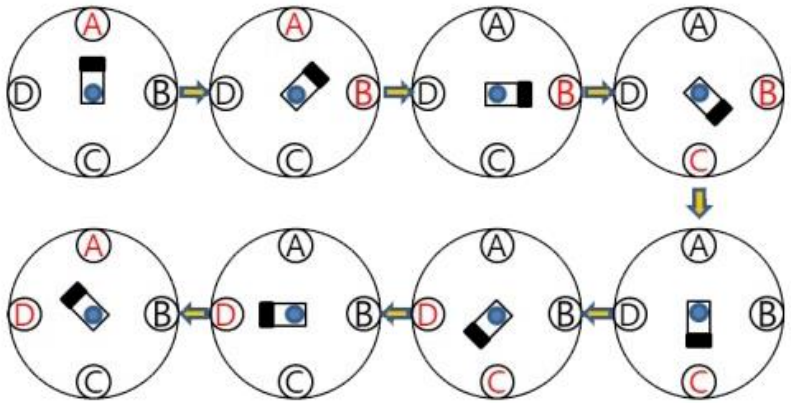
DC모터제어

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
MOTOR_P = 19
MOTOR_M = 13
If __name__ == “__main__”:
        GPIO.setup(MOTOR_P, GPIO.OUT)
        GPIO.setup(MOTOR_M, GPIO.OUT)
        try:
                while(True):
                 GPIO.output(MOTOR_P, GPIO.HIGH)
                 GPIO.output(MOTOR_M, GPIO.LOW)
                 print("Clock Wise")
                 time.sleep(1)
                 GPIO.output(MOTOR_P, GPIO.LOW)
                 GPIO.output(MOTOR_M, GPIO.LOW)
                 print("Stop")
                 time.sleep(1)
                 GPIO.output(MOTOR_P, GPIO.LOW)
                 GPIO.output(MOTOR_M, GPIO.HIGH)
                 print("Count Clock Wise")
                 time.sleep(1)
                 GPIO.output(MOTOR_P, GPIO.LOW)
                 GPIO.output(MOTOR_M, GPIO.LOW)
                 print("Stop")
                  time.sleep(1)
except KeyboardInterrupt:
#finally:
                GPIO.cleanup()
```

Step 모터제어

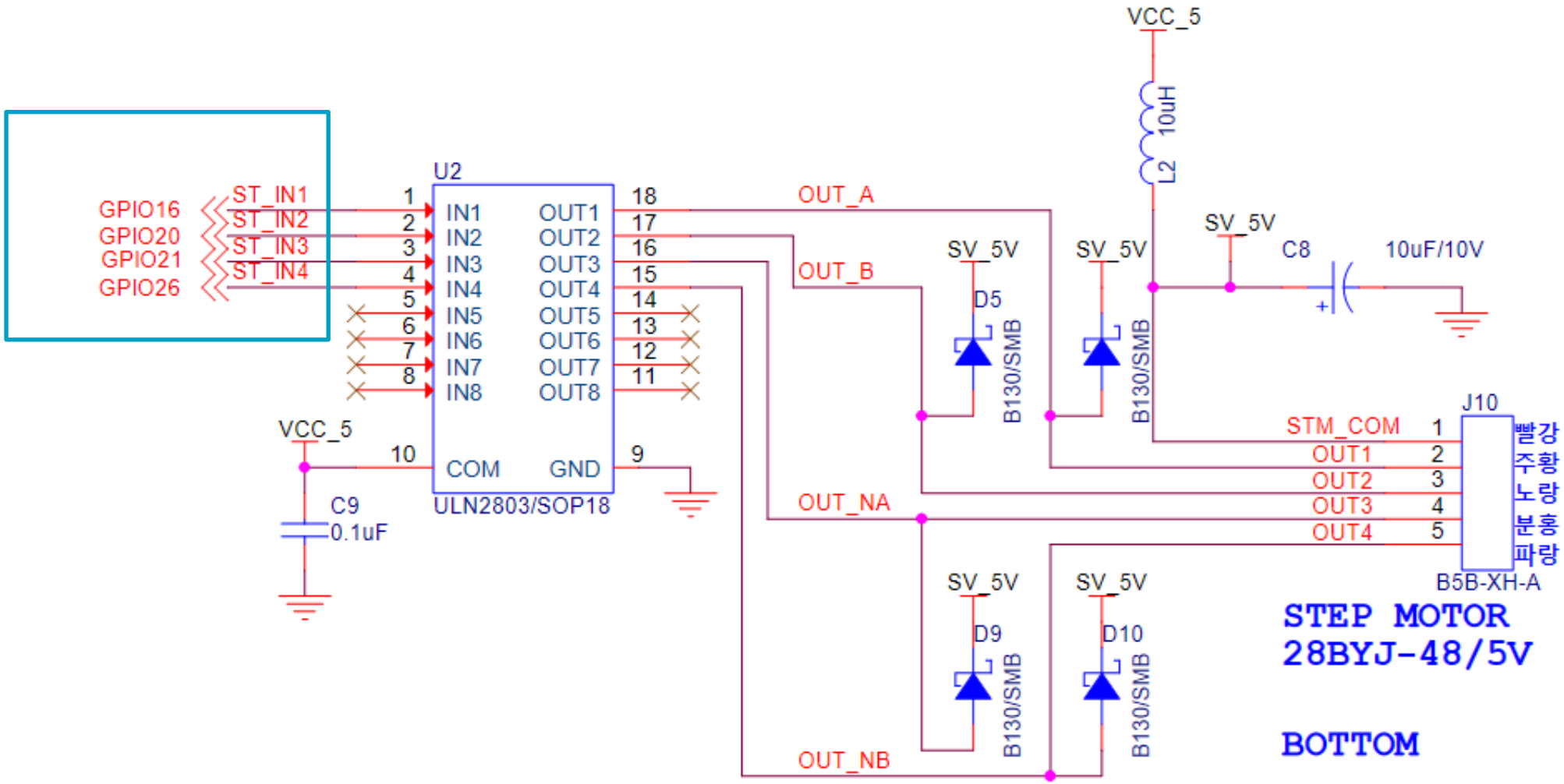






모터상	1	2	3	4	5	6	7	8
A (**A**) – in1	1	1	0	0	0	0	0	1
B (**B**) – in2	0	1	1	1	0	0	0	0
C (/**A**) – in3	0	0	0	1	1	1	0	0
D (/**B**) – in4	0	0	0	0	0	1	1	1

Step 모터제어


GPIO16
GPIO20
GPIO21
GPIO26
ST_IN1
ST_IN2
ST_IN3
ST_IN4
U2
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
COM
GND
ULN2803/SOP18
VCC_5
C9
0.1uF
OUT_A
OUT_B
OUT_NA
OUT_NB
SV_5V
D5
D9
D10
B130/SMB
L2 10uH
C8
10uF/10V
J10
STM_COM
OUT1
OUT2
OUT3
OUT4
빨강
주황
노랑
분홍
파랑
B5B-XH-A
STEP MOTOR
28BYJ-48/5V
BOTTOM

Step 모터제어-1

32 step = 360 degree, 4 pulses = 11.25 degree
Gearbox of 1:64

➔ 32*64 = 2048 steps for 360 degree rotation

```
import RPi.GPIO as GPIO
import time

# 스텝모터와 연결된 GPIO 번호를 변수에 저장한다.
STEP_IN1 = 16
STEP_IN2 = 20
STEP_IN3 = 21
STEP_IN4 = 26

pinsArray = [STEP_IN1,STEP_IN2,STEP_IN3,STEP_IN4]

# 풀 스탭 구동 (1상 여자 방식)
signal_full = [
        [GPIO.HIGH, GPIO.LOW, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.HIGH, GPIO.LOW, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.HIGH, GPIO.LOW],
        [GPIO.LOW, GPIO.LOW, GPIO.LOW, GPIO.HIGH]
        ]

# 1스탭의 사이클
FULL_STEP = len(pinsArray)
ROTATE_360_STEP = 512 # FULL_STEP으로 512스탭

if __name__ == "__main__":

    # BCM 핀맵을 사용한다
    GPIO.setmode(GPIO.BCM)
    for p_index in pinsArray:
        GPIO.setup(p_index, GPIO.OUT)
        GPIO.output(p_index, GPIO.LOW)
```

```
# 반복하여 모터를 정방향 -> 정지 -> 역방향 -> 정지 순서로 제어한다.
    try:
        for i in range(0,ROTATE_360_STEP):
            for step_idx in range(FULL_STEP):
                # GPIO를 제어한다. for문을 통해 코드를 줄일 수 있다.
                # GPIO.output(STEP_IN1, signal_full[step_idx][0])
                # GPIO.output(STEP_IN2, signal_full[step_idx][1])
                # GPIO.output(STEP_IN3, signal_full[step_idx][2])
                # GPIO.output(STEP_IN4, signal_full[step_idx][3])
                for idx in range(4):
                    GPIO.output(pinsArray[idx], signal_full[step_idx][idx])
                time.sleep(0.002)

        for i in range(0,ROTATE_360_STEP):
            for step_idx in reversed(range(FULL_STEP)):
                # GPIO를 제어한다. for문을 통해 코드를 줄일 수 있다.
                # GPIO.output(STEP_IN1, signal_full[step_idx][0])
                # GPIO.output(STEP_IN2, signal_full[step_idx][1])
                # GPIO.output(STEP_IN3, signal_full[step_idx][2])
                # GPIO.output(STEP_IN4, signal_full[step_idx][3])
                for idx in range(4):
                    GPIO.output(pinsArray[idx], signal_full[step_idx][idx])
                time.sleep(0.002)

    # 키보드 인터럽트, 에러 등으로 소스가 종료될 경우 GPIO를 초기화한 후 종료한다.
    finally:
        GPIO.cleanup()
```

Step 모터제어-2

```
import RPi.GPIO as GPIO
import time
from collections import deque


# 스텝모터와 연결된 GPIO 번호를 변수에 저장한다.
STEP_IN1 = 16
STEP_IN2 = 20
STEP_IN3 = 21
STEP_IN4 = 26



sig=deque([1,0,0,0])

step=2048

dir=1
```

```
if __name__ == "__main__":

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)


    GPIO.setup(STEP_IN1, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(STEP_IN2, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(STEP_IN3, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(STEP_IN4, GPIO.OUT, initial=GPIO.LOW)


    try:
        while(True):

            for cnt in range(0, step):
                GPIO.output(STEP_IN1,sig[0])
                GPIO.output(STEP_IN2,sig[1])
                GPIO.output(STEP_IN3,sig[2])
                GPIO.output(STEP_IN4,sig[3])
                time.sleep(0.01)
                sig.rotate(dir)

            dir=dir*-1

    except KeyboardInterrupt:
        GPIO.cleanup()
```

deque objects

class collections.**deque**([*iterable*[, *maxlen*]])

Returns a new deque object initialized left-to-right (using append()) with data from *iterable*. If *iterable* is not specified, the new deque is empty.

rotate(*n=1*)

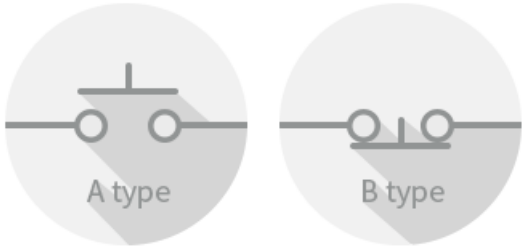
Rotate the deque *n* steps to the right. If *n* is negative, rotate to the left.

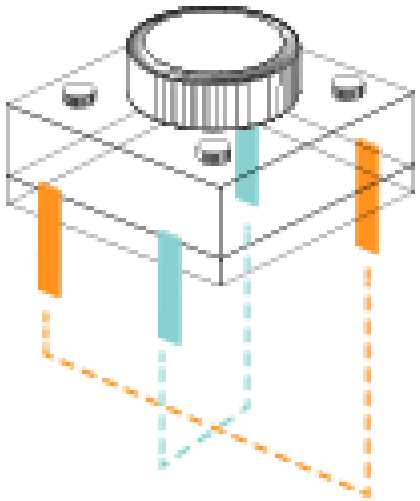
Step 모터제어-wiringpi

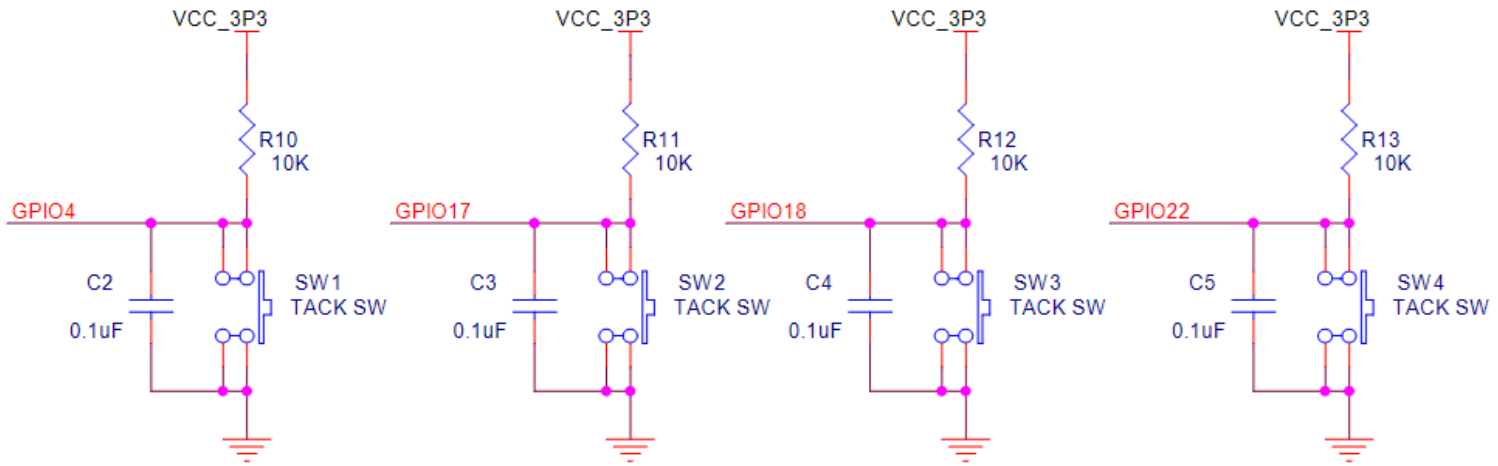
```c
#include <stdio.h>
#include <errno.h>
#include <string.h>

#include <wiringPi.h>
#define STEP_IN1  27
#define STEP_IN2  28
#define STEP_IN3  29
#define STEP_IN4  25


char pinsArray[4] = {STEP_IN1,STEP_IN2,STEP_IN3,STEP_IN4};

char signal_full[4][4] = {
          {1, 0, 0, 0},
          {0, 1, 0, 0},
          {0, 0, 1, 0},
          {0, 0, 0, 1}
          };

char FULL_STEP = 4;
int ROTATE_360_STEP = 512;

void setup_io(void){
        pinMode(STEP_IN1 , OUTPUT);
        pinMode(STEP_IN2 , OUTPUT);
        pinMode(STEP_IN3 , OUTPUT);
        pinMode(STEP_IN4 , OUTPUT);
        digitalWrite(STEP_IN1, LOW);
        digitalWrite(STEP_IN2, LOW);
        digitalWrite(STEP_IN3, LOW);
        digitalWrite(STEP_IN4, LOW);
}
```

```c
int main ()
{
      if (wiringPiSetup () == -1)
      {
            fprintf (stdout, "oops: %s₩n", strerror (errno)) ;
            return 1 ;
      }
      setup_io();

      for(int i = 0; i< ROTATE_360_STEP ; i++){
            for(char step_idx  = 0; step_idx  < FULL_STEP ; step_idx ++){
                  for(char idx = 0; idx < 4; idx++){
                  digitalWrite(pinsArray[idx], signal_full[step_idx ][idx]);

            }
            delay(1);
            }
      }

      for(int i = 0; i< ROTATE_360_STEP ; i++){
            for(char step_idx  = FULL_STEP ; step_idx  > 0 ; step_idx --){
                  for(char idx = 0; idx < 4; idx++){
                  digitalWrite(pinsArray[idx], signal_full[step_idx ][idx]);

                  }
                  delay(1);
            }
      }

      return 0;

}
```

스위치


A type
B type




VCC_3P3
R10
10K
GPIO4
C2
0.1uF
SW1
TACK SW
VCC_3P3
R11
10K
GPIO17
C3
0.1uF
SW2
TACK SW
VCC_3P3
R12
10K
GPIO18
C4
0.1uF
SW3
TACK SW
VCC_3P3
R13
10K
GPIO22
C5
0.1uF
SW4
TACK SW

Switch

```
import RPi.GPIO as GPIO
import time

# GPIO의 모드를 BCM으로 설정한다(GPIO번호 사용)
GPIO.setmode(GPIO.BCM)

SW1_PIN = 4
SW2_PIN = 17
SW3_PIN = 18
SW4_PIN = 22

SW_PIN_LIST = [SW1_PIN, SW2_PIN, SW3_PIN, SW4_PIN]
```

```
if __name__ == "__main__":

    # GPIO 핀을 INPUT 으로 설정한다.
    GPIO.setup(SW_PIN_LIST, GPIO.IN)

    try:
        while(True):

            button_state = []

            for i in SW_PIN_LIST:
                button_state.append(GPIO.input(i))


            print("Button State : ", button_state)

            # 0.5초간 대기한다.
            time.sleep(0.5)

    finally:
        GPIO.cleanup()
```

Switch-1

```c
#include <stdio.h>
#include <string.h>

#include <wiringPi.h>
#define SW1_PIN  7
#define SW2_PIN  0
#define SW3_PIN  1
#define SW4_PIN  3


void setup_io(void){
 pinMode(SW1_PIN, INPUT);
 pinMode(SW2_PIN, INPUT);
 pinMode(SW3_PIN, INPUT);
 pinMode(SW4_PIN, INPUT);
}
```

```c
int main ()
{
    if (wiringPiSetup () == -1)
    {
        fprintf (stdout, "oops: %s₩n", strerror
        (errno)) ;
        return 1 ;
    }
    setup_io();

    for (;;) {
        printf("Button State: %d %d %d %d ₩n",
        digitalRead(SW1_PIN), digitalRead(SW2_PIN),
        digitalRead(SW3_PIN), digitalRead(SW4_PIN));
    }

    return 0;

}
```

과제

SW1 누름시 DC모터 정회전
SW2 누름시 DC모터 역회전
SW3 누름시 DC모터 정지

* 정회전 및 역회전을 시작 하기전 모터를 정지하고 구동하시오

ftp 서버 설치

sudo apt-get install vsftpd
sudo nano /etc/vsftpd.conf

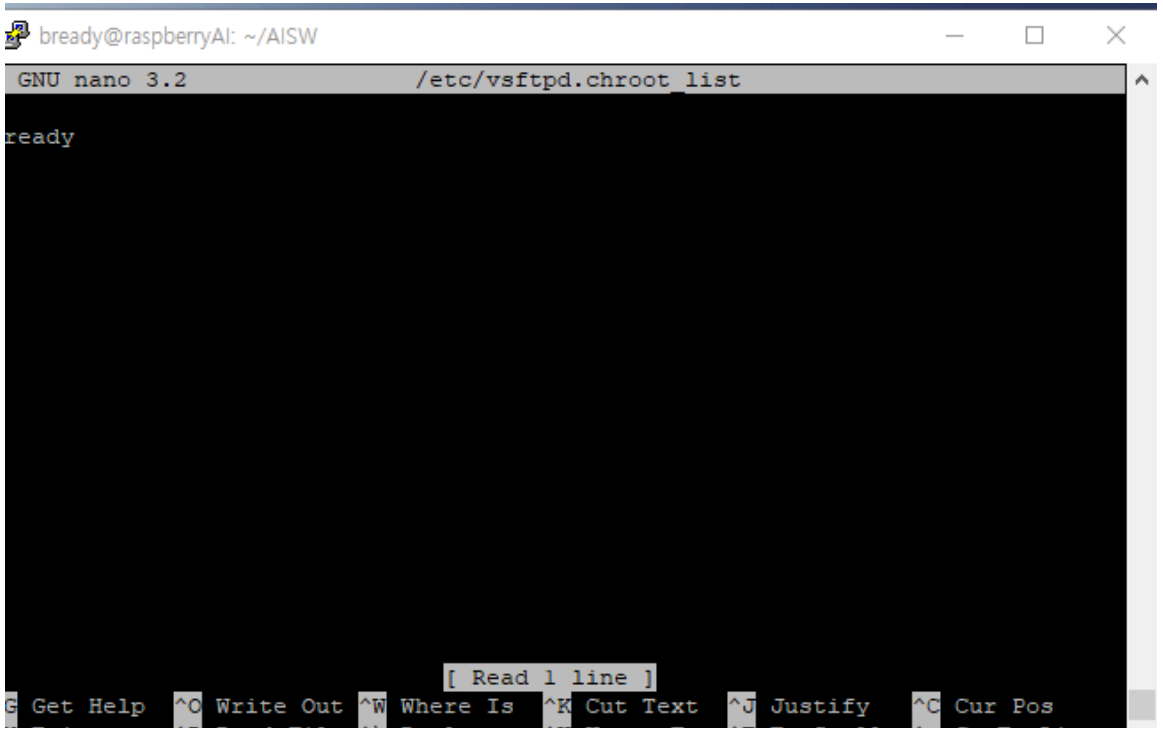
#내용중 주석 삭제
local_enable=YES
write_enable=YES
local_umask=022

chroot_local_user = YES
chroot_list_enable = YES
chroot_list_file=/etc/vsftpd.chroot_list

아이디 저장
sudo nano /etc/vsftpd.chroot_list
아이디는 bready 로 입력후 저장

재시작
sudo systemctl restart vsftpd

자동실행
sudo systemctl enable vsftpd



ftp 클라이언트 설치

₩₩192.168.24.251
접속하여 filezilla 설치